OPENFABRICS ALLIANCE

Microsoft SMB 2.2 - Running Over RDMA in Windows Server “8”

Tom Talpey, Architect
Microsoft
March 27, 2012

SMB2 Background

- The primary Windows filesharing protocol
- Initially shipped in Vista and Server 2008
  - Simplified command set (as compared to SMB1)
  - Uniformity (UNICODE, timestamps, etc.)
  - Expanded identifier space (UINT64)
  - HMAC-SHA256 signing
  - Dynamic crediting
  - Asynchronous notifications for long running requests
  - Unrestricted compounding of requests
  - Symbolic Link support
  - Durable opens for handling disconnects
- Updated in Windows 7 and Server 2008R2
  - Frame reduction for common workloads and WAN
    - SMB2 Leasing
    - Branch Cache extensions
  - Large MTU support (increases throughput)
  - Resilient Handles

SMB2.2 – Under Development

- Extensively enhanced for Windows 8
  - Multichannel
  - Encryption
  - SMB2 over RDMA
  - Persistent Handles
  - Scale-Out Awareness
  - Witness Notification Protocol
  - Clustered Client Failover
  - Directory Oplocks
  - Branch Cache v2
- Enables SMB2 use by new server applications

SMB2.2 Fileservers

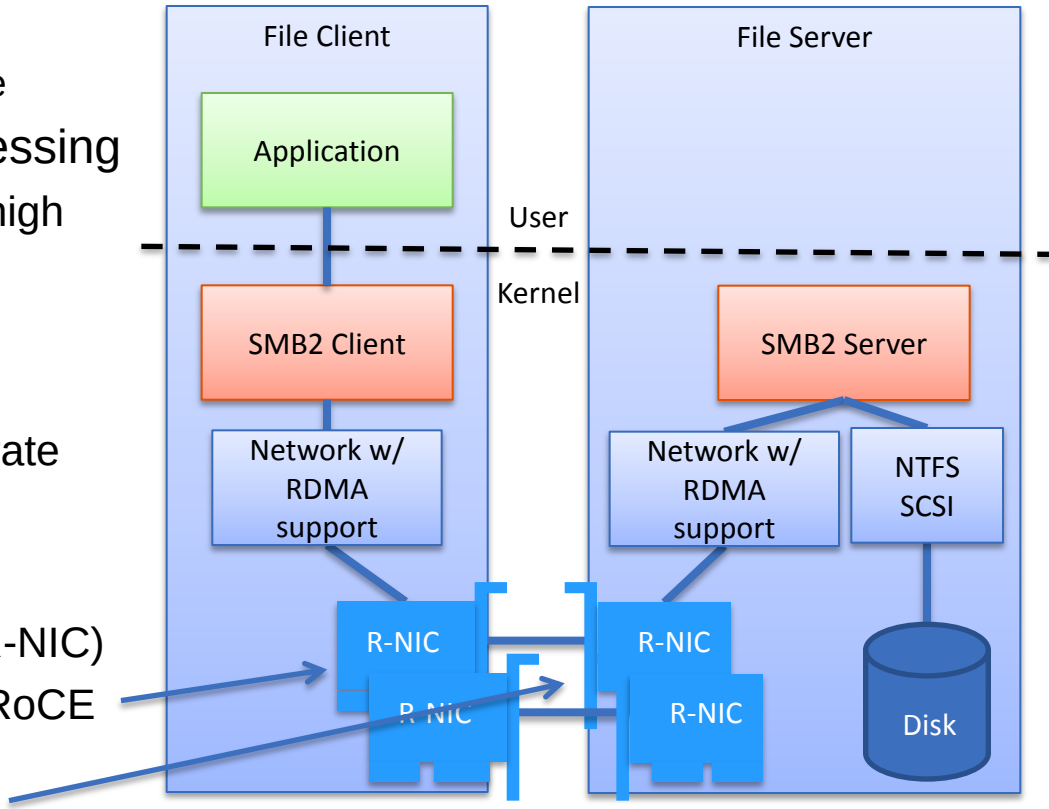
- Are fully supported to store and serve:
  - Client files
  - Hyper-V Virtual Hard Disks and configuration
  - SQL server data
  - …and more
- Are “Continuously Available”
  - Zero downtime with node and network fault tolerance
  - Transparent failover to SMB2.2 clients
  - Planned and unplanned failover
- Are highly performing
  - Using multiple TCP and now RDMA channels

OPENFABRICS ALLIANCE

SMB Direct

SMB Direct (SMB2 over RDMA)

- A new RDMA-enabled transport for SMB2.2
  - Enables a new class of SMB2 file storage
- Minimal CPU utilization for I/O processing
  - Low latency and ability to leverage high speed NICs
- Traditional advantages of SMB2 file storage
  - Easy to provision, manage and migrate
  - Leverages converged network
- Required hardware
  - RDMA-capable network interface (R-NIC)
  - Support for iWARP, InfiniBand and RoCE
- Works with SMB2 Multichannel for Discovery, Load Balancing/Failover



Goals of SMB Direct

- Enable new class of SMB2 file storage for server apps
  - Minimal client-side CPU utilization for file storage processing
  - Low latency and ability to leverage high speed NICs
- Remote file storage similar to local in functionality, performance, reliability, availability
- No application change
- No administrator configuration
- Transparently fall-forward and fall-back on changes of RDMA and TCP connectivity
- Strict data integrity

NDKPI

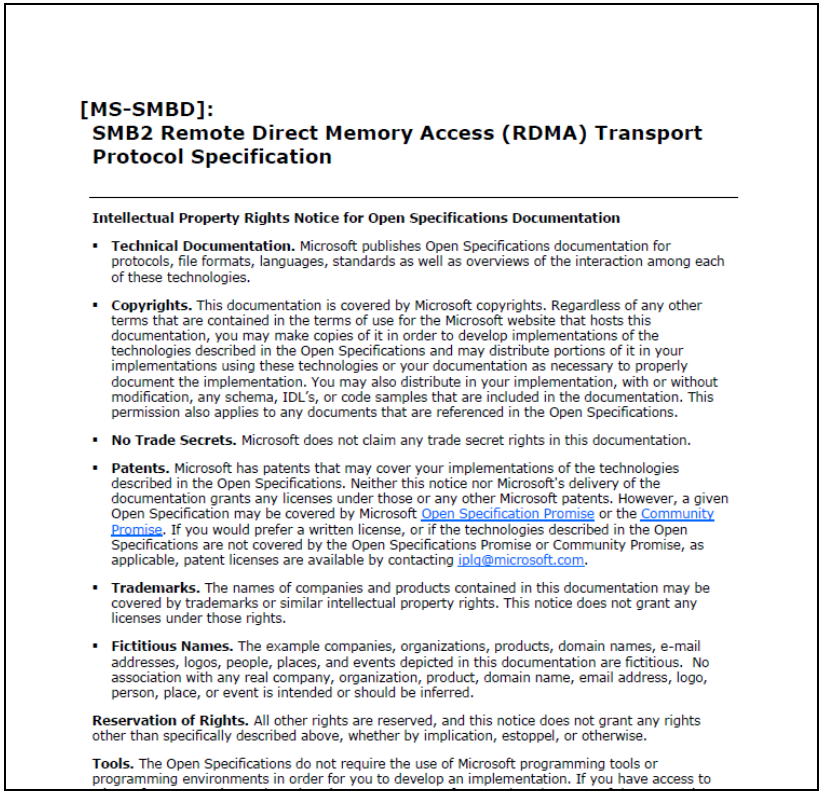
- Network Direct Kernel Programming Interface
- New RDMA API for Windows Server "8"
  - Architectural approach similar to NDSPI
- Equally supports multiple RDMA Providers
  - iWARP
  - InfiniBand
  - RoCE
- Asynchronous, Windows kernel-optimized API
- http://msdn.microsoft.com/en-us/library/windows/hardware/hh463974(v=vs.85).aspx

Open Protocol Documentation

- All SMB2 protocols published, including RDMA
  - http://www.microsoft.com/protocols
- SMB2.2 Protocol Family
  - Existing docs updated
    - MS-SMB2 – SMB2 (all dialects)
    - MS-DFSC – DFS Namespaces
    - MS-FSCC – File System Control Codes
    - MS-FSA – File System Algorithms
  - New Protocol Documents
    - **MS-SMBD – SMB Direct (RDMA)**
    - MS-FSRVP – Remote VSS Protocol
    - MS-SWN – SMB Witness Protocol

SMB Direct Protocol Specification

- New document
  - **MS-SMBD**
- Sits “below” MS-SMB2 in the SMB2 stack
  - Transport framing layer
    - Peer to Direct TCP/445



[MS-SMBD]:
SMB2 Remote Direct Memory Access (RDMA) Transport Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to

- Available from Microsoft protocol documentation:
  - http://msdn.microsoft.com/en-us/library/hh536346(v=prot.13).aspx

SMB Direct use of RDMA

- The SMB2.2 client **directs** all use of RDMA
  - For SMB2 Reads and Writes only
- The SMB2.2 server **performs** all RDMA
  - Improves security, integrity and performance
- Zero-copy, zero-touch
  - Buffer cache use is supported optionally on both

SMB Direct use of RDMA …

- Uses a simple RDMA profile
  - Fabric agnostic
  - Any memory registration type
  - No optional features required
    - E.g. atomics, remote invalidate, etc
- No shared PDs
- No memory region caching
- Strict invalidation after RDMA use

Relationship to NFS/RDMA

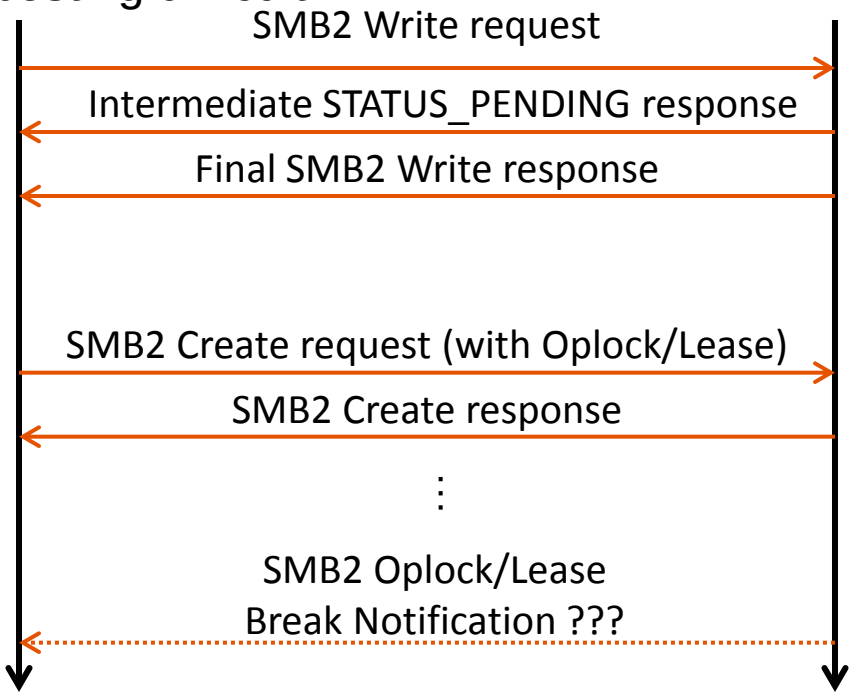
- A very different approach!
- NFS/RDMA defines an RPC transport for NFS
  - RPC is strict request/response – SMB2 is not
  - NFS has well-defined request/response sizes – SMB2 does not
- NFS/RDMA does not expose RDMA to NFS
  - NFS operations are unmodified – SMB2.2 read and write optionally carry RDMA information
- Result – SMB Direct is a very simple lower layer
  - Efficient and flexible

OPENFABRICS ALLIANCE

Protocol details

RDMA Technical Challenges

The SMB2 protocol is pre-existing. Unpredictable SMB2 responses, and unacknowledged SMB2 requests, make RDMA crediting and pre-posting difficult.



Example 1

SMB2 requests that go async resulting in two responses from the server.

Example 2

Oplocks/leases break notifications may never be received if the oplock/lease isn't broken by another client.

Additionally, SMB2 requests and responses have a highly variable size (from tens of bytes up to 1 megabyte and more)

SMB Direct Protocol

- These challenges shape the protocol:
- Just three SMB Direct protocol messages
  - Negotiate Request
  - Negotiate Response
  - Data Transfer
- Credits indicate when it is safe to send a packet to the peer and how many sends may be performed
- Fragmentation is used to transmit non-RDMA messages that are larger than the negotiated MTU
- Two data transfer modes
  - **Send/Receive mode** used to transmit SMB2 metadata requests and small SMB2 reads/writes (typically <8KB)
  - **RDMA mode** used to transmit data for large SMB2 reads/writes

SMB Direct Credits

- SMB Direct protocol uses credits to control flow of SMB Direct Data Transfer messages
  - Different from, and in addition to, SMB2 layer credits
- Each credit represents a pre-posted, fixed-size receive and entitles the credit receiver to perform one send to the credit granter
- Credits are granted bi-directionally and asymmetrically between the client and server with every message that they exchange
  - *CreditsRequested* - total number of credits that the message sender wants to have (including the credits they already have)
  - *CreditsGranted* - number of additional credits granted to the message recipient

Connection Establishment

- **MinVersion/MaxVersion**– Range of protocol versions (inclusive) supported by the sender. Currently only 0x0100/0x0100.
- **CreditsRequested** – Used to implement flow control.
- **PreferredSendSize** – the number of bytes that the sender requests to be able to transmit to the receiver via a single SMB Direct Data Transfer message.
- **MaxReceiveSize**– the maximum number of bytes that the sender is willing to receive via a single SMB2 Data Transfer message.
- **MaxFragmentedReceiveSize** – size, in bytes, of the largest fragmented upper-layer message that can be received by the sender.
- *Note, IRD/ORD are provided by the transport.*

SMB Direct Negotiate Request

Octet 0	Octet 1	Octet 2	Octet 3
MinVersion		MaxVersion	
Reserved		CreditsRequested	
PreferredSendSize			
MaxReceiveSize			
MaxFragmentedReceiveSize			

The SMB Direct Negotiate Request message is the first message sent by the active host once the RDMA transport level connection has been established.

Connection Establishment…

- **MinVersion/MaxVersion**– Range of protocol versions (inclusive) supported by the sender. Currently only 0x0100/0x0100.
- **NegotiatedVersion** – Selected protocol version
- **CreditsRequested /CreditsGranted** – Used to implement flow control.
- **Status** – Connection negotiation success/failure
- **MaxReadWriteSize** – Maximum size, in bytes, that the sender will RDMA Read/Write from/to the client per upper-layer request.
- **PreferredSendSize** – the number of bytes that the sender requests to be able to transmit to the receiver via a single SMB Direct Data Transfer message.
- **MaxReceiveSize**– the maximum number of bytes that the sender is willing to receive via a single SMB Direct Data Transfer message.
- **MaxFragmentedReceiveSize** – size, in bytes, of the largest fragmented upper-layer message that can be received by the sender.

SMB Direct Negotiate Response

Octet 0	Octet 1	Octet 2	Octet 3
MinVersion		MaxVersion	
NegotiatedVersion		Reserved	
CreditsRequested		CreditsGranted	
Status			
MaxReadWriteSize			
PreferredSendSize			
MaxReceiveSize			
MaxFragmentedReceiveSize			

The SMB Direct Negotiate Response message is the first message sent by the passive peer in response to the active peer's SMB Direct Negotiate Request.

Send/Receive Data Transfer

- **Credits Requested/Granted**– flow control as negotiated.
- **Flags**– 0 or KEEPALIVE_REQUESTED (0x0001).
- **RemainingDataLength**– used to fragment and transmit data payloads that are larger than the peer's max receive size.
- **DataOffset**– the offset, in bytes, from the beginning of the header to the data payload.
- **DataLength** – the length, in bytes, of the data payload.
- **Data** – the payload (usually an SMB2 message but may be empty). Padded to 8-byte alignment.

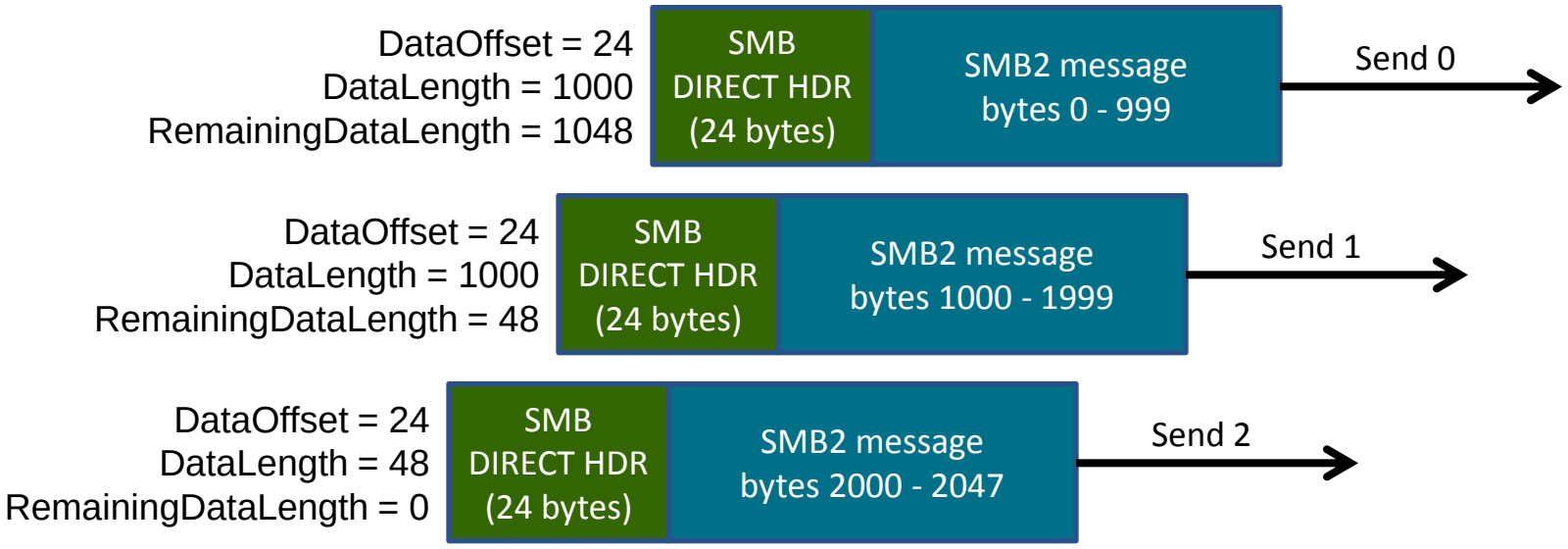
SMB Direct Data Transfer Header

Octet 0	Octet 1	Octet 2	Octet 3
CreditsRequested		CreditsGranted	
Flags		Reserved	
RemainingDataLength			
DataOffset			
DataLength			
Padding			
Data (variable)			

All subsequent messages are Data Transfers. Empty data transfer messages may also be exchanged as required for keepalive and credit management.

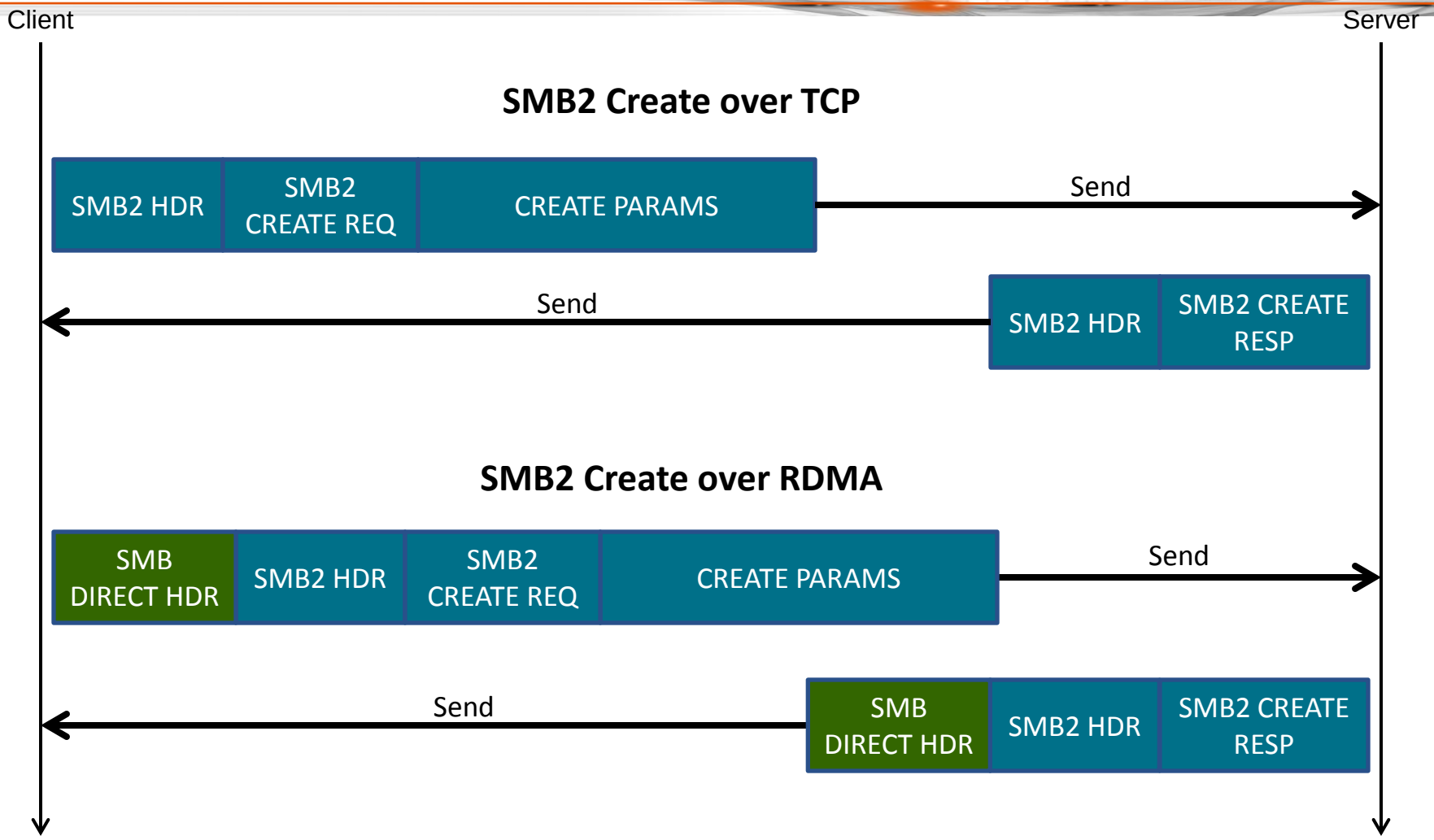
Send/Receive Fragmentation

Transmitting a 2K SMB2 message when the peer's *MaxReceiveSize* is only 1K

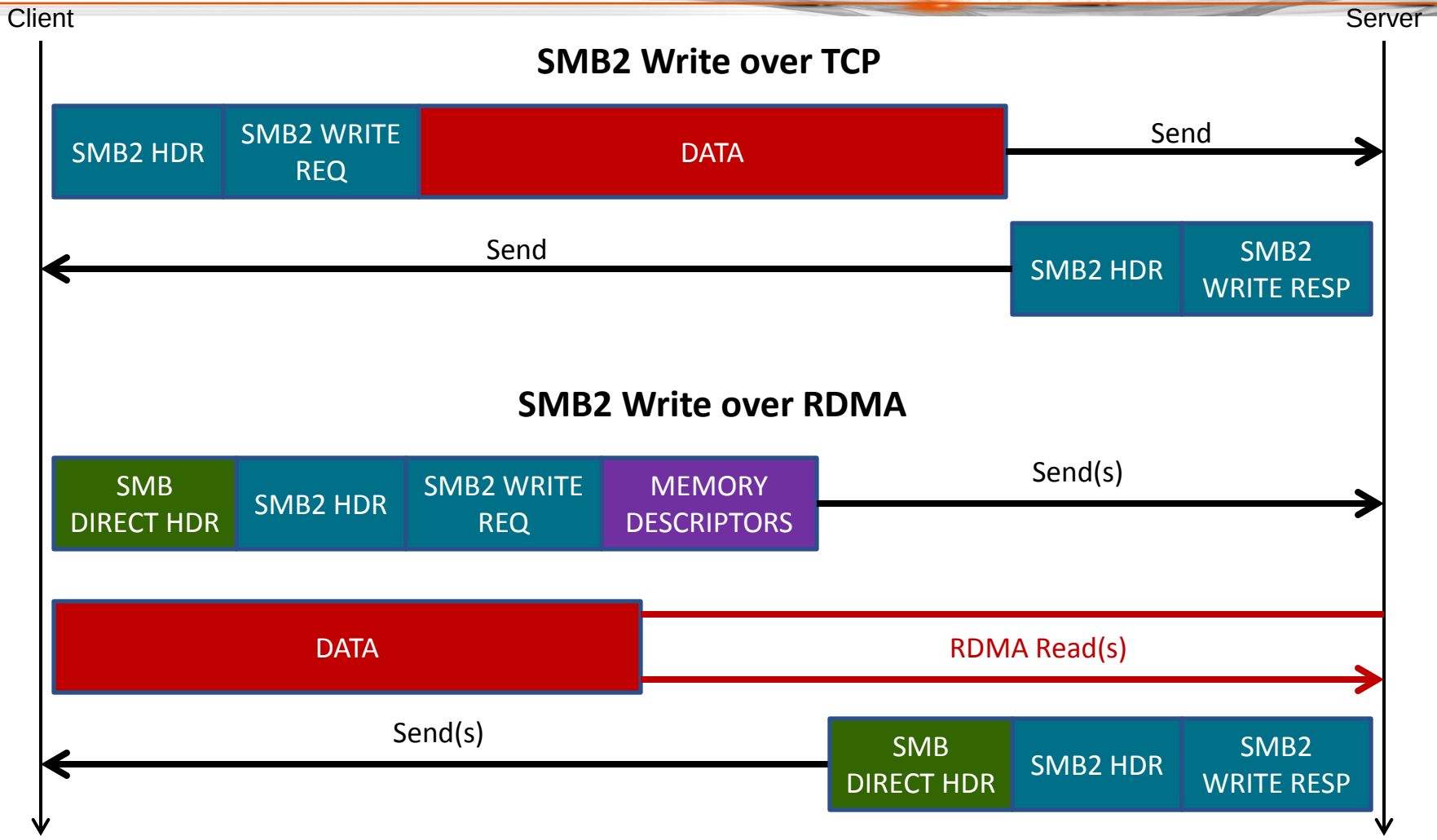


- Entire SMB2 message is received when *RemainingDataLength* == 0
- Total size of SMB2 message is indicated by the first fragment (*DataLength* + *RemainingDataLength*) which is <= receiver's *MaxFragmentedReceiveSize*
- Payload fragments are transmitted sequentially. The protocol relies on RDMA's strong ordering guarantees to handle fragments in the correct order.

SMB2 Traffic (Send/Receive mode)



Large SMB2 Write (RDMA mode)



Large SMB2 Write (RDMA mode)…

- **DataOffset** – ignored by server when Channel is non-zero.
- **Channel** – set to 0x1 to identify the channel info contents as V1 memory descriptors.
- **WriteChannelInfoOffset** – the offset, in bytes, from the beginning of the SMB2 header to the memory descriptor array.
- **WriteChannelInfoLength** – the length, in bytes, of the memory descriptor array.
- **Buffer** – The memory descriptor array as described by *WriteChannelInfoOffset* and *WriteChannelInfoLength.* Each array element consists of:

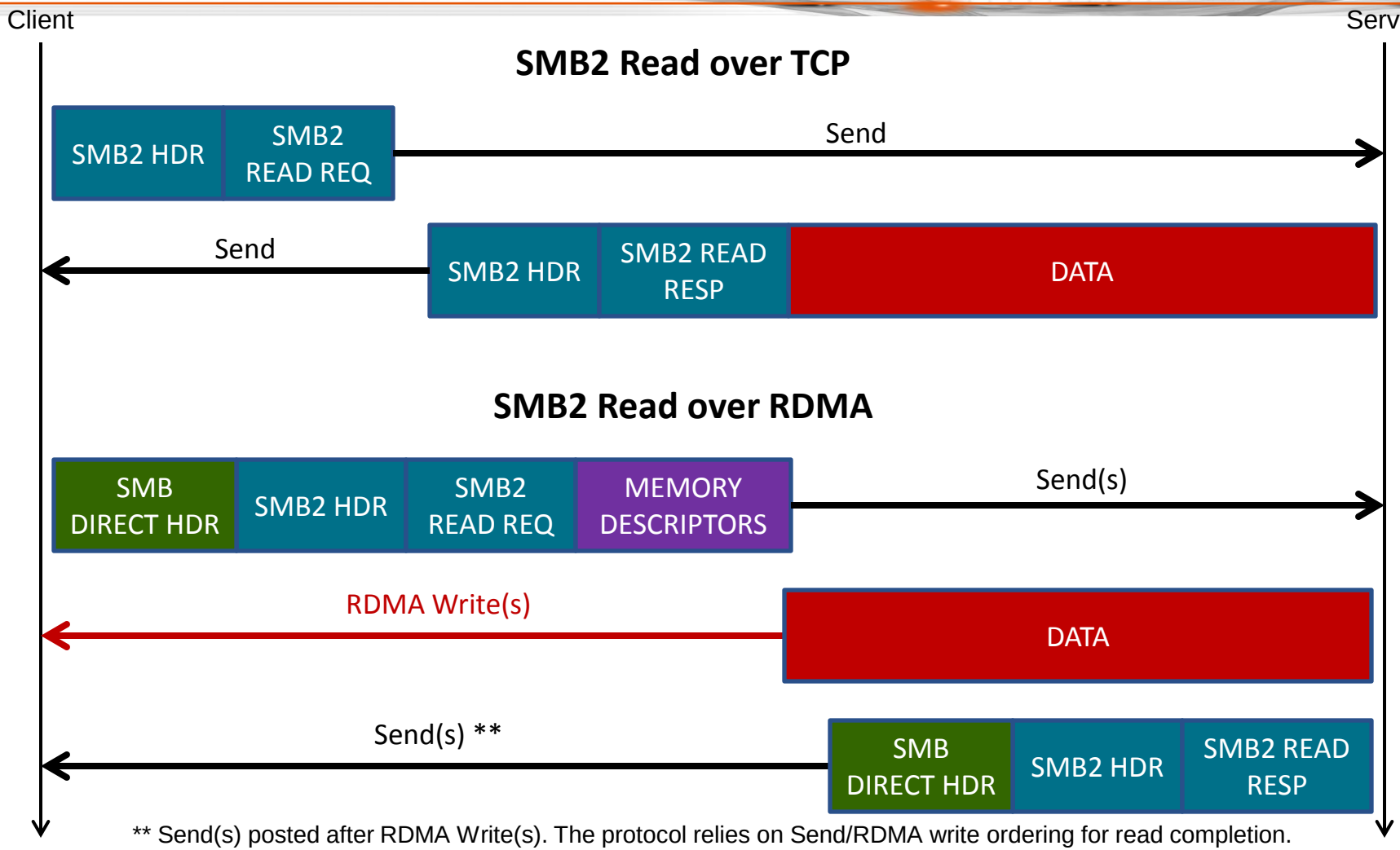
Octet 0	Octet 1	Octet 2	Octet 3
Address			
…			
Token			
Length			

SMB2 WRITE REQUEST

Octet 0	Octet 1	Octet 2	Octet 3
StructureSize		DataOffset	
Length			
Offset			
…			
FileId			
…			
…			
…			
Channel			
RemainingBytes			
WriteChannelInfoOffset		WriteChannelInfoLength	
Flags			
Buffer (variable)			

Previously reserved field

Large SMB2 Read (RDMA mode)



** Send(s) posted after RDMA Write(s). The protocol relies on Send/RDMA write ordering for read completion.

Large SMB2 Reads (RDMA mode)…

- **Channel** – set to 0x1 to identify the channel info contents as V1 memory descriptors.
- **ReadChannelInfoOffset** – the offset, in bytes, from the beginning of the SMB2 header to the memory descriptor array.
- **ReadChannelInfoLength** – the length, in bytes, of the memory descriptor array.
- **Buffer** – The memory descriptor array as described by *ReadChannelInfoOffset* and *ReadChannelInfoLength.* Each array element consists of:

Octet 0	Octet 1	Octet 2	Octet 3
Address			
…			
Token			
Length			

SMB2 READ REQUEST

Octet 0	Octet 1	Octet 2	Octet 3
StructureSize		Padding	Reserved
Length			
Offset			
…			
FileId			
…			
…			
…			
MinimumCount			
Channel			
RemainingBytes			
ReadChannelInfoOffset		ReadChannelInfoLength	
Flags			
Buffer (variable)			

Previously reserved field

OPENFABRICS ALLIANCE

Summary

SMB Direct is…

- A simple, efficient protocol
- Highly adaptable to the many ~~quirks~~ needs of upper layers
- An RDMA provider-agnostic transport layer
- With NDKPI, able to support new providers, fabrics, etc.

What you need

- Windows Server “8” developer preview
  - http://www.microsoft.com/en-us/server-cloud/windows-server/v8-default.aspx
- Currently-supported RNICs:
  - Intel NetEffect NE020, 10GbE
  - Mellanox ConnectX2, IB QDR / RoCE 10GbE
  - Mellanox ConnectX3, IB FDR / RoCE 40GbE
- Jose Barreto’s blog a great source for setup:
  - http://blogs.technet.com/b/josebda/
  - http://blogs.technet.com/b/josebda/archive/2012/03/15/windows-server-8-beta-scale-out-file-server-for-sql-server-2012-step-by-step-installation.aspx

Performance results

- Bandwidth: Excellent (high)
- IOPS: Excellent (high)
- Overhead: Excellent (low)
- Scalable ×N using multiple connections (!)

- Watch this space… ☺

Questions ?